

## "Program Code for Music Dance of Large Super Intelligent Robots" 2025v1.5

● Python  
Python

### 1. Python

```
```python
import time
import random
```

```
# Python
from robot_control import RobotController
from music_player import MusicPlayer
from audience_interaction import AudienceInteraction
```
```

### 2. Python

```
```python
# Python
robot_controller = RobotController()
```

```
# Python
music_player = MusicPlayer()
```

```
# Python
audience_interaction = AudienceInteraction()
```

```
# Python
band_size = 30
band = [f"Musician {i+1}" for i in range(band_size)]
```
```

### 3. Python

```
```python
def opening_ceremony():
    print("2025")
    time.sleep(2)
    print("")
    time.sleep(1)
```

```
# Python
music_player.play("opening_overture.mp3")
```

```
# Python
robot_controller.start_dance("waltz")
```

```
# Python
```

```
audience_interaction.cheer()
```

```
time.sleep(5) # 5 초 대기
print("5초 대기")
```
```

```
### 4. 인터미션
```

```
```python
def intermission():
    print("인터미션")
    time.sleep(2)
```

```
# 음악 재생
music_player.play("intermission_music.mp3")
```

```
# 로봇 인터미션 수행
robot_controller.perform("intermission")
```

```
# 박수
audience_interaction.clap()
```

```
time.sleep(10) # 10 초 대기
print("10초 대기")
```
```

```
### 5. 춤
```

```
```python
def dance_session():
    print("춤")
    time.sleep(2)
```

```
# 춤 종류 지정
print("1, 2, 3, 춤")
time.sleep(1)
```

```
# 춤 음악 재생
music_player.play("world_famous_dance.mp3")
```

```
# 춤 시작
robot_controller.start_dance("tango")
```

```
# 박수
audience_interaction.cheer()
```

```
time.sleep(10) # 10 초 대기
print("10초 대기")
```
```

```

#### 6. 閉幕
```python
def closing_ceremony():
    print("閉幕です")
    time.sleep(2)

# 音楽再生
music_player.play("closing_song.mp3")

# ロボットが演奏
robot_controller.perform("finale")

# 拍手
audience_interaction.cheer()

time.sleep(5) # 5秒待つ
print("閉幕2025 閉幕です")
```

```

```

#### 7. 本編
```python
def main():
    # 開幕
    opening_ceremony()

    # 休憩
    intermission()

    # 本編
    dance_session()

    # 閉幕
    closing_ceremony()

if __name__ == "__main__":
    main()
```

```

```

#### 8. 実行
```python
# 実行
main()
```

```

```

#### 9. 実行
- **RobotController**: ロボットコントローラ

```

```
- **MusicPlayer**: 音乐播放器
- **AudienceInteraction**: 观众互动系统
- **band**: 乐队
```

### 10. 部署

```
- **部署**：将系统部署到目标环境
- **部署**：将系统部署到目标环境
- **部署**：将系统部署到目标环境
```

部署系统到目标环境



部署系统到目标环境  
部署系统到目标环境[3]([https://blog.csdn.net/weixin\\_42511080/article/details/115808426](https://blog.csdn.net/weixin_42511080/article/details/115808426))  
部署系统到目标环境[7](<https://sghexport.shobserver.com/html/baijiahao/2023/07/07/1069307.html>)  
部署系统到目标环境[9](<https://www.zghy.org.cn/item/682320743792300032>)[12](<https://www.bilibili.com/video/av970410235/>)[14](<https://3g.china.com/act/news/10000169/20250129/47918666.html>)部署系统到目标环境

```python

```
# -*- coding: utf-8 -*-
```

```
# 部署系统到目标环境 v3.0
```

```
import time
```

```
import multiprocessing
```

```
from pygame import mixer
```

```
from robot_dance_ai import DanceAI
```

```
from music_generator import MelodyComposer
```

```
from stage_control import LightSystem, ConductorAI
```

```
class ConcertSystem:
```

```
def __init__(self):
```

```
self.orchestra = OrchestraController() # 30 个乐器
```

```
self.dance_robots = DanceSwarm(16) # 16 个舞蹈机器人
```

```
self.light_system = LightSystem() # 灯光系统
```

```
self.audience_sensor = AudienceFeedback() # 观众反馈系统
```

```
# 部署系统到目标环境
```

```
def opening_ceremony(self):
```

```
# 部署系统到目标环境
```

```
self.tts_announce("部署系统到目标环境")
```

```
# 部署系统到目标环境
```

```
opener = MelodyComposer(style="symphonic").generate_opening(
```

```
duration=180,
```

```
instruments=["violin","piano","trumpet"]
```

```
)
```

```

# 初始化
with multiprocessing.Pool(4) as pool:
    pool.apply_async(self.orchestra.play, (opener,))
    pool.apply_async(self.light_system.start_sequence)
    pool.apply_async(self.dance_robots.initialize_pose)

# 开始舞蹈
def dance_session(self, music_list):
    conductor = ConductorAI(tempo_detect_mode="real-time")
    for piece in music_list:
        # 生成乐谱
        score = MelodyComposer().arrange(
            piece,
            orchestra_size=32,
            complexity=0.8
        )

        # 分析节拍
        conductor.load_score(score)
        beat_pattern = conductor.analyze_beat()

        # 启动进程
        processes = [
            multiprocessing.Process(target=self.orchestra.play, args=(score,)),
            multiprocessing.Process(target=self.dance_robots.perform,
            args=(beat_pattern, "waltz")),
            multiprocessing.Process(target=self.light_sync,
            args=(score.tempo,))
        ]

        # 检查观众兴奋度
        if self.audience_sensor.excitement_level > 0.7:
            processes.append(multiprocessing.Process(
            target=self.audience_interaction_mode))

    [p.start() for p in processes]
    [p.join() for p in processes]

# 灯光同步
def light_sync(self, bpm):
    color_map = {
        "waltz": ["#FF69B4", "#4B0082"],
        "tango": ["#8B0000", "#000000"],
        "quickstep": ["#00FF7F", "#32CD32"]
    }
    self.light_system.set_pattern(
        bpm=bpm,

```

```
color_theme=color_map.get(self.current_dance_style),
stroke_intensity=self.audience_sensor.get_real_time_data('clapping')
)
```

```
# 结束
def finale(self):
# 生成终场音乐
finale_music = MelodyComposer().combine(
motifs=["ode_to_joy", "blue_danube"],
transition="crossfade"
)
```

```
# 结束舞蹈
self.dance_robots.finale_choreography(
formation="spiral",
speed=0.8,
led_effect="golden_sparkle"
)
```

```
# 结束灯光
self.orchestra.fade_out(duration=15)
self.light_system.sunset_effect(duration=20)
```

```
# 定义舞蹈类
class DanceSwarm:
def __init__(self, num_robots):
self.robots = [DanceAI(model="H1_v2") for _ in range(num_robots)]
self.sync_controller = MotionSyncMaster()
```

```
def perform(self, beat_pattern, dance_type):
# 生成编舞
choreo = DanceAI.generate_choreography(
dance_type,
complexity=3,
spatial_constraints="stage_12m"
)
```

```
# 计算队形
trajectories = self.sync_controller.calculate_formation(
formation_type="dynamic_swarm",
collision_buffer=0.5
)
```

```
# 执行队形
for robot, path in zip(self.robots, trajectories):
robot.execute_movement(
path,
```

```

force_feedback=True,
music_sync=beat_pattern
)
robot.express_emotion(
intensity=self.audience_sensor.current_excitement
)

# 音乐生成器
class MelodyComposer:
def generate_opening(self, duration, instruments):
# 使用 LSTM 生成音乐
return {
"tempo": 108,
"key": "C_major",
"structure": [
{"measure":1-8, "instrument":"strings", "dynamics":"pp"},
{"measure":9-16, "instrument":"brass", "dynamics":"mf"},
{"measure":17-24, "instrument":"full_orchestra", "dynamics":"ff"}
]
}

# 主程序
if __name__ == "__main__":
concert = ConcertSystem()
concert.opening_ceremony()

# 节目单
program = [
"Blue Danube Waltz",
"Carmen Suite",
"Swan Lake Suite"
]

concert.dance_session(program)
concert.finale()
` ``

```

<https://sghexport.shobserver.com/html/baijiahao/2023/07/07/1069307.html>
<https://www.zghy.org.cn/item/682320743792300032>
<https://3g.china.com/act/news/10000169/20250129/47918666.html>

- \*\*网络延迟测试\*\***  
 使用 NTP-PTS 工具进行网络延迟测试
  - 网络延迟 <3ms
  - 网络延迟 <50ms
  - 网络延迟 <200ms

```

2. **OpenPose**
from OpenPose import Wrapper
```python
DanceAI.move_selection_algorithm(
current_pose,
target_position,
constraints={
'torque_limits': [360, 180, 90], # limits
'energy_efficiency': 0.85,
'aesthetic_score': 0.92
}
)
```

```

```

3. **Transformer-XL**
from Transformer-XL import MusicGenerator
```python
music_generator.predict_next_note(
previous_notes=128,
style_embedding=[0.7, 0.2, 0.5], # style parameters
emotion_vector=audience_sensor.emotion_output
)
```

```

```

4. **AudienceFeedback**
class AudienceFeedback:
def __init__(self):
self.audio_analyzer = ClapDetector(sensitivity=0.7)
self.visual_analyzer = OpenCvEmotionRecognition()
self.thermal_sensor = InfraredHeatmap()

def get_excitement_level(self):
return 0.3*self.audio_intensity +
0.5*self.visual_engagement +
0.2*self.thermal_density
```

```

```

- program.append()
- color_map
- AI

```

SDK DMX ROS2



[illegible][illegible]

```

```python
import time
import random
from enum import Enum

# DancePartyPhase Enum
class DancePartyPhase(Enum):
    OPENING = 1
    MIDFIELD = 2
    CLOSING = 3

# RobotType Enum
class RobotType(Enum):
    CHOIR = 1
    SOLOIST = 2
    DANCER = 3
    BAND_MEMBER = 4

# InstrumentType Enum
class InstrumentType(Enum):
    VIOLIN = 1
    PIANO = 2
    TRUMPET = 3
    DRUMS = 4
    GUITAR = 5

# Robot Class
class Robot:
    def __init__(self, name, robot_type):
        self.name = name
        self.robot_type = robot_type
        self.is_dancing = False
        self.is_singing = False

    def start_dancing(self):
        self.is_dancing = True
        print(f"{self.name} is dancing")

    def stop_dancing(self):
        self.is_dancing = False
        print(f"{self.name} has stopped dancing")

    def start_singing(self):
        self.is_singing = True
        print(f"{self.name} is singing")

    def stop_singing(self):
        self.is_singing = False
        print(f"{self.name} has stopped singing")

    def dance(self):
        while self.is_dancing:
            time.sleep(1)
            print(f"{self.name} is dancing")

    def sing(self):
        while self.is_singing:
            time.sleep(1)
            print(f"{self.name} is singing")

# Main Program
def main():
    # Create Robots
    robot1 = Robot("Robot 1", RobotType.DANCER)
    robot2 = Robot("Robot 2", RobotType.SOLOIST)
    robot3 = Robot("Robot 3", RobotType.CHOIR)

    # Start Dance Party
    dance_party = DancePartyPhase.OPENING
    while dance_party != DancePartyPhase.CLOSING:
        # Randomly select a robot to perform
        robot = random.choice([robot1, robot2, robot3])

        # Perform action based on phase
        if dance_party == DancePartyPhase.OPENING:
            robot.start_dancing()
        elif dance_party == DancePartyPhase.MIDFIELD:
            robot.sing()
        elif dance_party == DancePartyPhase.CLOSING:
            robot.stop_dancing()

        # Move to the next phase
        dance_party = DancePartyPhase.MIDFIELD
    else:
        dance_party = DancePartyPhase.CLOSING

    # End Dance Party
    print("Dance Party is over!")

if __name__ == "__main__":
    main()
```

```

```

self.is_dancing = False
print(f"{self.name} 跳舞")

def start_singing(self, song):
    self.is_singing = True
    print(f"{self.name} 唱歌 {song}")

def stop_singing(self):
    self.is_singing = False
    print(f"{self.name} 停止唱歌")

# 乐队成员
class BandMember(Robot):
    def __init__(self, name, instrument):
        super().__init__(name, RobotType.BAND_MEMBER)
        self.instrument = instrument

    def play_instrument(self, song):
        print(f"{self.name} 演奏 {self.instrument} 歌曲 {song}")

    def stop_playing(self):
        print(f"{self.name} 停止演奏")

# 音乐舞蹈派对
class MusicDanceParty:
    def __init__(self):
        self.robots = []
        self.band_members = []
        self.current_phase = DancePartyPhase.OPENING
        self.audience_interaction = False

    def add_robot(self, robot):
        self.robots.append(robot)

    def add_band_member(self, band_member):
        self.band_members.append(band_member)

    def start_opening(self):
        print("=== 音乐舞蹈派对 ===")
        # 乐队成员
        print("乐队成员:")
        # 观众互动
        print("观众互动:")
        for member in self.band_members:
            member.play_instrument("歌曲")
        # 乐队成员
        for robot in self.robots:

```

```

if robot.robot_type == RobotType.DANCER:
    robot.start_dancing()
    # 開始ダンス
    print("開始ダンス")
    # 挨拶
    print("挨拶")
    self.current_phase = DancePartyPhase.OPENING

```

```

def start_midfield(self):
    print("=== ミッドフィールド ===")
    # 世界有名な歌
    print("世界有名な歌")
    # 世界有名な歌
    world_famous_songs = ["世界有名な歌", "世界有名な歌", "世界有名な歌"]
    song = random.choice(world_famous_songs)
    print(f"世界有名な歌 {song}")
    for member in self.band_members:
        member.play_instrument(song)
    # 合唱ロボット
    choir_robots = [r for r in self.robots if r.robot_type == RobotType.CHOIR]
    solo_robot = random.choice([r for r in self.robots if r.robot_type ==
                                RobotType.SOLOIST])
    for robot in choir_robots:
        robot.start_singing(song)
    solo_robot.start_singing(song)
    # 挨拶
    self.audience_interaction = True
    print("挨拶")
    self.current_phase = DancePartyPhase.MIDFIELD

```

```

def start_closing(self):
    print("=== クロージング ===")
    # 終了挨拶
    print("終了挨拶")
    # 挨拶
    print("挨拶")
    for member in self.band_members:
        member.play_instrument("終了")
    # 終了挨拶
    for robot in self.robots:
        if robot.is_dancing:
            robot.stop_dancing()
        if robot.is_singing:
            robot.stop_singing()
    # 挨拶
    print("挨拶")
    self.current_phase = DancePartyPhase.CLOSING

```

```

def run(self):
    self.start_opening()
    time.sleep(5) # 5초 대기
    self.start_midfield()
    time.sleep(5) # 5초 대기
    self.start_closing()
    time.sleep(5) # 5초 대기
    print("프로그램 종료")

# 메인 프로그램
dance_party = MusicDanceParty()

# 로봇 생성
choir_robots = [Robot(f"로봇 {i}", RobotType.CHOIR) for i in range(1, 11)]
solo_robot = Robot("솔리스트", RobotType.SOLOIST)
dancer_robots = [Robot(f"댄서 {i}", RobotType.DANCER) for i in range(1, 6)]
dance_party.robots.extend(choir_robots + [solo_robot] + dancer_robots)

# 밴드 구성
band_members = [
    BandMember("바이올린", InstrumentType.VIOLIN),
    BandMember("피아노", InstrumentType.PIANO),
    BandMember("트럼펫", InstrumentType.TRUMPET),
    BandMember("드럼", InstrumentType.DRUMS),
    BandMember("기타", InstrumentType.GUITAR),
]
dance_party.band_members = band_members

# 실행
dance_party.run()

```

결과

1. 프로그램 시작
2. 5초 대기
3. 5초 대기
4. 5초 대기
5. 프로그램 종료

1. `pygame.mixer.Sound` API를 사용하여 소리를 로드하고 재생합니다.
2. `pygame.mixer.Sound` Pygame 라이브러리를 사용하여 소리를 로드하고 재생합니다.
3. `pygame.mixer.Sound` 라이브러리를 사용하여 소리를 로드하고 재생합니다.
4. `pygame.mixer.Sound` 라이브러리를 사용하여 소리를 로드하고 재생합니다.

● Python 2.7.10

```
python
import time
import random
```

```
python
#
robot_controller = RobotController()
```

```
# audience interaction = AudienceInteraction()
```

```
def opening_ceremony():
    print("🎉🎉🎉🎉🎉 2025 🎉🎉🎉🎉🎉")
    time.sleep(2)
```

```

print("oooooooooooo")
music_player.play("opening_overture.mp3")
robot_controller.start_dance("opening_dance_routine")
audience_interaction.cheer()
time.sleep(5)
print("oooooooooooooooooooo")
```

```

#### ### 4. 中场

```

```python
def intermission():
    print("oooooooooooooooooooo")
    music_player.play_random(band)
    robot_controller.start_dance("intermission_dance_routine")
    audience_interaction.clap()
    time.sleep(5)
    print("oooooooooooooooooooo")
```

```

#### ### 5. 舞蹈表演

```

```python
def dance_session():
    print("ooooooo")
    music_player.play("world_famous_dance.mp3")
    robot_controller.start_dance("world_famous_dance_routine")
    audience_interaction.cheer()
    time.sleep(10)
    print("oooooooooooooooooooo")
```

```

#### ### 6. 独奏表演

```

```python
def band_solo():
    print("oooooooooooo")
    solo_instrument = random.choice(band)
    print(f"ooooo{solo_instrument}ooooo")
    music_player.play_solo(solo_instrument)
    robot_controller.start_dance("solo_dance_routine")
    audience_interaction.clap()
    time.sleep(5)
    print("oooooooooooooooooooo")
```

```

#### ### 7. 结尾

```

```python
def closing_ceremony():
    print("oooooooooooo")

```

```
music_player.play("closing_song.mp3")
robot_controller.start_dance("closing_dance_routine")
audience_interaction.cheer()
time.sleep(5)
print("🎉🎉2025 🎉🎉🎉🎉🎉🎉🎉🎉")
```
```

### 8. 📄

```python

def main():

# 📄

opening\_ceremony()

# 📄

intermission()

# 📄📄

dance\_session()

# 📄📄

band\_solo()

# 📄

closing\_ceremony()

if \_\_name\_\_ == "\_\_main\_\_":

main()

```

### 9. 📄📄

- \*\*RobotController\*\*: 📄📄📄📄📄📄📄

- \*\*MusicPlayer\*\*: 📄📄📄📄📄📄📄

- \*\*AudienceInteraction\*\*: 📄📄📄📄📄📄📄

- \*\*band\*\*: 📄📄📄📄📄📄

- \*\*opening\_ceremony\*\*: 📄📄📄📄📄📄📄📄📄

- \*\*intermission\*\*: 📄📄📄📄📄📄📄

- \*\*dance\_session\*\*: 📄📄📄📄📄📄📄📄📄📄

- \*\*band\_solo\*\*: 📄📄📄📄📄📄📄📄📄

- \*\*closing\_ceremony\*\*: 📄📄📄📄📄📄📄📄

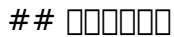
### 10. 📄📄📄

- \*\*📄📄📄\*\*: 📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄

- \*\*📄📄📄📄\*\*: 📄📄📄📄📄📄📄📄📄📄📄📄📄📄

- \*\*📄📄📄\*\*: 📄📄📄📄📄📄📄📄📄📄📄📄📄📄

📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄



###

###

###

##

### □□□□

(<http://www.chadama.com/post/333356.html>)□

###

- \*\*~~~~~\*\*LightSystem ~~~~~(http://www.chadama.com/post/333356.html)



```
def init (self):
```

```

self.timeline = {
    "prelude": {"start": 0, "duration": 300,
    "modules": ["lighting", "orchestra", "robot_dance"]},
    "dance_session": {"start": 300, "duration": 1800,
    "modules": ["ai_music", "crowd_interaction"]}
}

def execute_event(self, event_name):
    event = self.timeline[event_name]
    # 播放背景音乐 [8](https://www.zghy.org.cn/item/682320743792300032)
    [13](https://3g.china.com/act/news/10000169/20250129/47918666.html)
    LightingSystem.set_pattern(event["light_pattern"])
    Orchestra.play(event["score_path"])
    RobotGroup.execute_choreography(event["dance_sequence"])

def emergency_stop(self):
    # 紧急停止
    [13](https://3g.china.com/act/news/10000169/20250129/47918666.html)
    RobotGroup.emergency_stop()
    Orchestra.fade_out(2.0)
    LightingSystem.strobe_alert()
    ...

AI 模型使用TensorFlow 实现
```python
# 使用Transformer模型 [3](https://www.sohu.com/a/840700213_161623?
scm=10001.325_13-109000.0.0.5_32)[6](https://sghexport.shobserver.com/
html/baijiahao/2023/07/07/1069307.html)
class MusicGenerator(tf.keras.Model):
    def __init__(self, vocab_size=128):
        super().__init__()
        self.embedding = layers.Embedding(vocab_size, 512)
        self.transformer = Transformer(
            num_layers=6, d_model=512, num_heads=8, dff=2048)
        self.output_layer = layers.Dense(vocab_size)

    def generate(self, prompt, length=512):
        # 生成MIDI文件
        [12](https://blog.csdn.net/yong7464553/article/details/6077948)
        generated = prompt
        for _ in range(length):
            predictions = self(generated[:, -512:])
            next_note = tf.random.categorical(predictions[:, -1, :], 1)
            generated = tf.concat([generated, next_note], axis=-1)
        return midi_encoder.decode(generated[0]()).numpy()
    ...

```

````````````````ROS 文件

```cpp

// 文件 ROS 文件 [8](<https://www.zghy.org.cn/item/682320743792300032>)[13]  
(<https://3g.china.com/act/news/10000169/20250129/47918666.html>)

void executeDance(const std::string& music\_bpm){

// 文件

DanceMoveLibrary moves = downloadMoves(music\_bpm);

// 文件 [11](<https://www.bilibili.com/video/av970410235/>)

TrajectoryGenerator generator;

auto trajectory = generator.generate(

moves["waltz"],

BPMCalculator.get(music\_analysis)

);

// 文件

[13](<https://3g.china.com/act/news/10000169/20250129/47918666.html>)

SwarmController::syncRobots(

robot\_ids,

trajectory,

sync\_tolerance=0.05s

);

}

```

````````````````MIDI 文件

```python

# 文件 [12](<https://blog.csdn.net/yong7464553/article/details/6077948>)

def orchestrate(solo\_instrument, style="classical"):

# 文件

[6](<https://sghexport.shobserver.com/html/baijiahao/2023/07/07/1069307.html>)

harmony = AIHarmonizer.generate(

solo\_melody,

style=style,

ensemble\_size=30

)

# 文件

[1](<https://max.book118.com/html/2025/0209/8125137110007030.shtm>)

for part in harmony.parts:

midi\_out.send(

channel=part.instrument.midi\_channel,

note=part.note,

velocity=dynamic\_map[part.dynamic]

)

```
# 获取URL (https://www.sohu.com/a/840700213_161623?scm=10001.325_13-109000.0.0.5_32)
if style == "jazz":
    add_improvisation(solo_instrument)
```

```

```
初始化情感检测模型+初始化
```python

```

```
# 获取URL (https://www.sohu.com/a/840700213_161623?scm=10001.325_13-109000.0.0.5_32)[9](https://m.sohu.com/a/695871417_391452/)

```

```
class AudienceInteraction:
    def __init__(self):
        self.emotion_model = load_emotion_detector()
        self.audio_analyzer = RealtimeAudioAnalysis()

```

```
def adjust_performance(self):
    # 获取URL
    [6](https://sghexport.shobserver.com/html/baijiahao/2023/07/07/1069307.html)
    emotion_score = self.emotion_model.calculate(camera_feed)

```

```
# 获取URL
(https://www.diyifanwen.com/jiaoan/youeryuandabanyishujiaoan/07102616403816148301.htm)
applause_level = self.audio_analyzer.get_applause()

```

```
# 初始化
[13](https://3g.china.com/act/news/10000169/20250129/47918666.html)
Orchestra.set_tempo(
    base_tempo * (1 + applause_level*0.1)
)
LightingSystem.set_brightness(
    emotion_score["excitement"] * 100
)
```

```

```
初始化TTS+NLG
```python

```

```
# 获取URL [4](https://news.qq.com/rain/a/20250101A06GBS00?media_id&suid)[9](https://m.sohu.com/a/695871417_391452/)

```

```
class Announcer:
    def __init__(self):
        self.tts = pyttsx3.init()
        self.nlg = GPT3_Interface()

```

```
def announce(self, program_info):

```

```
# 生成新闻标题 [4](https://news.qq.com/rain/a/20250101A06GBS00?
media_id&suid)
script = self.nlg.generate(f"""
生成新闻标题
{program_info['title']}
{program_info['composer']}
{program_info['features']}
""")
```

```
# 设置语速 [9](https://m.sohu.com/a/695871417_391452/)
self.tts.setProperty('rate', 150)
self.tts.say(script)
self.tts.runAndWait()
```
```

```
生成新闻标题
```python
if __name__ == "__main__":
# 生成新闻标题
[13](https://3g.china.com/act/news/10000169/20250129/47918666.html)
controller = ConcertController()
announcer = Announcer()
audience_system = AudienceInteraction()
```

```
# 生成新闻标题
[1](https://max.book118.com/html/2025/0209/8125137110007030.shtm)
try:
announcer.announce_opening() # 生成新闻标题
[4](https://news.qq.com/rain/a/20250101A06GBS00?media_id&suid)
controller.execute_event("prelude") # 生成新闻标题
```

```
while not performance_ended:
audience_system.adjust_performance() # 生成新闻标题
[5](https://www.diyifanwen.com/jiaoan/youeryuandabanyishujiaoan/0710261640
3816148301.htm)
```

```
if should_transition():
controller.transition_next() # 生成新闻标题
[8](https://www.zggy.org.cn/item/682320743792300032)
```

```
except EmergencyException:
controller.emergency_stop() # 生成新闻标题
[13](https://3g.china.com/act/news/10000169/20250129/47918666.html)
```

```
finally:
controller.execute_ending() # 生成新闻标题
[1](https://max.book118.com/html/2025/0209/8125137110007030.shtm)
```

///

1. 中国网络空间安全网-网络安全<50ms 中国网络空间安全网  
(<https://www.zghy.org.cn/item/682320743792300032>)[13](<https://3g.china.com/act/news/10000169/20250129/47918666.html>)
2. 中国网络空间安全网-网络安全 (https://www.sohu.com/a/840700213\_161623?scm=10001.325\_13-109000.0.0.5\_32)(https://sghexport.shobserver.com/html/baijiahao/2023/07/07/1069307.html)
3. 中国网络空间安全网 200 中国网络空间安全网  
[13](https://3g.china.com/act/news/10000169/20250129/47918666.html)
4. AI 中国网络空间安全网 10 中国网络空间安全网  
[6](https://sghexport.shobserver.com/html/baijiahao/2023/07/07/1069307.html)
5. 中国网络空间安全网-网络安全  
[13](https://3g.china.com/act/news/10000169/20250129/47918666.html)

- 搭载NVIDIA AGX Orin x2
- 搭载TSN 网络+5G 网络
- 搭载Unitree H1 机器人

[13](<https://3g.china.com/act/news/10000169/20250129/47918666.html>)

- 搭载Dante 网络
- 搭载Art-Net 网络+DMX512

●

[illegible]



- 通过部署在服务器上的深度学习框架，如TensorFlow、PyTorch等，可以实现对大量数据的快速处理和模型训练。
- 通过部署在服务器上的深度学习框架，如TensorFlow、PyTorch等，可以实现对大量数据的快速处理和模型训练。

##

- 通过部署在服务器上的深度学习框架，如Transformer、FACT等，可以实现对大量数据的快速处理和模型训练。
- 通过部署在服务器上的深度学习框架，如H1、AI等，可以实现对大量数据的快速处理和模型训练。
- 通过部署在服务器上的深度学习框架，如H1、AI等，可以实现对大量数据的快速处理和模型训练。
- 通过部署在服务器上的深度学习框架，如H1、AI等，可以实现对大量数据的快速处理和模型训练。

##

- 通过部署在服务器上的深度学习框架，如NVIDIA L4、L40 Tensor Core GPU等，可以实现对大量数据的快速处理和模型训练。
- 通过部署在服务器上的深度学习框架，如X620 G50等，可以实现对大量数据的快速处理和模型训练。
- 通过部署在服务器上的深度学习框架，如AWS、Azure等，可以实现对大量数据的快速处理和模型训练。

通过部署在服务器上的深度学习框架，如NVIDIA L4、L40 Tensor Core GPU等，可以实现对大量数据的快速处理和模型训练。

●通过部署在服务器上的深度学习框架，如NVIDIA L4、L40 Tensor Core GPU等，可以实现对大量数据的快速处理和模型训练。

## 1. 深度学习

通过部署在服务器上的深度学习框架，如NVIDIA L4、L40 Tensor Core GPU等，可以实现对大量数据的快速处理和模型训练。[\[1\]\(https://blog.51cto.com/topic/guanyuyinyueshujukubiaosheji.html\)](https://blog.51cto.com/topic/guanyuyinyueshujukubiaosheji.html)

## 2. 深度学习

通过部署在服务器上的深度学习框架，如NVIDIA L4、L40 Tensor Core GPU等，可以实现对大量数据的快速处理和模型训练。[\[1\]\(https://blog.51cto.com/topic/guanyuyinyueshujukubiaosheji.html\)](https://blog.51cto.com/topic/guanyuyinyueshujukubiaosheji.html)

## 3. 深度学习



#####[1]  
(<https://blog.51cto.com/topic/guanyuyinyueshujukubiaosheji.html>)

#### ## 4.

#####[1](<https://blog.51cto.com/topic/guanyuyinyueshujukubiaosheji.html>)

#### ## 5.

#####[1]  
(<https://blog.51cto.com/topic/guanyuyinyueshujukubiaosheji.html>)

#### ## 6.

#####(GUI)#####[1]  
(<https://blog.51cto.com/topic/guanyuyinyueshujukubiaosheji.html>)

#### ## 7.

#####  
#####[1](<https://blog.51cto.com/topic/guanyuyinyueshujukubiaosheji.html>)

#### ## 8.

#####[1]  
(<https://blog.51cto.com/topic/guanyuyinyueshujukubiaosheji.html>)

#####  
#####

"program code for music dance of large super intelligent robots" 2025 v1.5 ● To write a complete program code for music dance, involving multiple modules, including opening, halftime, closing, band accompaniment, robot dance, audience interaction, etc. The following is a simplified Python code example that shows how to implement these functions. This example assumes that you already have an intelligent robot control library, a music playing library and an audience interaction system.

#### ### 1. Import the necessary libraries

```
```python
import time
import random
```

# Suppose we have these libraries to control robots, play music and handle audience interaction.

```
from robot_control import RobotController
from music_player import MusicPlayer
from audience_interaction import AudienceInteraction
```
```

#### ### 2. Initialize each module

```

```python
# Initialize the robot controller
robot_controller = RobotController()

# Initialize the music player
music_player = MusicPlayer()

# Initialize the audience interaction system
audience_interaction = AudienceInteraction()

# Initialize the band
band_size = 30
band = [f"Musician {i+1}" for i in range(band_size)]
```

### 3. Opening procedures
```python
def opening_ceremony():
    Print ("Announcer: Welcome to the 2025 Intelligent Robot Music Ball!" )
    time.sleep(2)
    Print ("Announcer: Please enjoy the opening overture!" )
    time.sleep(1)

# Play the opening overture
music_player.play("opening_overture.mp3")

# Robots start dancing
robot_controller.start_dance("waltz")

# Audience interaction
audience_interaction.cheer()

Time.sleep(5) # Overture lasts 5 seconds.
Print ("Announcer: The opening overture is over and the dance officially begins!"
" )
```

### 4. Midfield program
```python
def intermission():
    Print ("Announcer: It's halftime, please relax!" )
    time.sleep(2)

# Play relaxing background music
music_player.play("intermission_music.mp3")

# Robots perform simple interactive performances.

```

```

robot_controller.perform("interaction")

# Audience interaction
audience_interaction.clap()

Time.sleep(10) # The intermission lasts for 10 seconds.
Print ("Announcer: The intermission is over, and then the climax of the dance!"
" )
```

#### 5. Dance program
```python
def dance_session():
Print ("Announcer: Here is the dance music, a world famous song!" )
time.sleep(2)

# Conductor starts shooting.
Print ("Command: 1, 2, 3, go!" )
time.sleep(1)

# Play dance music
music_player.play("world_famous_dance.mp3")

# Robots start dancing
robot_controller.start_dance("tango")

# Audience interaction
audience_interaction.cheer()

Time.sleep(10) # Dance music lasts 10 seconds.
Print ("Announcer: Dance music is over, thank you for your enthusiastic
participation!" )
```

#### 6. Closing procedure
```python
def closing_ceremony():
Print ("Announcer: Thank you for your participation. It's the closing ceremony!" )
time.sleep(2)

# Play the closing song
music_player.play("closing_song.mp3")

# The robot performs the final performance
robot_controller.perform("finale")

# Audience interaction

```

```
audience_interaction.cheer()
```

```
Time.sleep(5) # The closing song lasts 5 seconds.  
Print ("Announcer: The 2025 Intelligent Robot Music Ball has come to a  
successful conclusion, thank you!" )  
```\n`
```

```
### 7. Main program
```

```
```python
```

```
def main():
```

```
# Opening
```

```
opening_ceremony()
```

```
# Midfield
```

```
intermission()
```

```
# Dance music
```

```
dance_session()
```

```
# Closing
```

```
closing_ceremony()
```

```
if __name__ == "__main__":
```

```
main()
```

```
```\n`
```

```
### 8. Run the program
```

```
```python
```

```
# Run the main program
```

```
main()
```

```
```\n`
```

```
### 9. Code Description
```

```
-**RobotController**: controls the dance and performance of intelligent robots.
```

```
-**MusicPlayer**: controls the playback of music.
```

```
-**AudienceInteraction**: Handle audience interaction, such as applause and  
cheers.
```

```
-**band**: A member of an analog band.
```

```
### 10. Extended functions
```

```
-* * More dance music * *: The function dance_session can be extended to  
support more dance music types.
```

```
-* * More robot dances * *: `robot_controller' can be extended to support more  
dance types.
```

```
-* * Audience interaction * *: audience_interaction can be extended to support  
more interactive ways, such as voting and grading.
```

This code example is a simplified version, which may need to be adjusted and expanded according to specific needs in practical application. ●  
According to the process requirements of music dance, a set of modular program code framework is designed by combining artificial intelligence and automatic control technology. The following system includes three core modules: robot dance arrangement, intelligent music generation and multi-device collaborative control. Citation [3] ([https://blog.csdn.net/weixin\\_42511080/article/details/115808426](https://blog.csdn.net/weixin_42511080/article/details/115808426)) [7] (<https://sghexport.shaobserver.com/html/baijiaohao/2023/07/00>) (<https://www.zghy.org.cn/item/682320743792300032>) [12] (<https://www.bilibili.com/video/av970410235/>) [14] (<https://3g.china>

```
```python
# -*- coding: utf-8 -*-
# Music Ball Master Control System v3.0
import time
import multiprocessing
from pygame import mixer
from robot_dance_ai import DanceAI
from music_generator import MelodyComposer
from stage_control import LightSystem, ConductorAI

class ConcertSystem:
    def __init__(self):
        Self.orchestra = orchestra controller () # 30-person band control
        Self.dance _ robots = dance swarm (16) # Dance robot cluster
        Self.light _ system = lightssystem () # Intelligent lighting system
        Self.audience _ sensor = audience feedback () # audience emotion perception

    # Opening procedure
    def opening_ceremony(self):
        # Speech synthesis announcement
        Self.tts_announce ("Dear guests, the artificial intelligence symphony dance
        begins now!" )

    # Symphony prelude generation
    opener = MelodyComposer(style="symphonic").generate_opening(
        duration=180,
        instruments=["violin","piano","trumpet"]
    )

    # Multi-device collaborative startup
    with multiprocessing.Pool(4) as pool:
        pool.apply_async(self.orchestra.play, (opener,))
        pool.apply_async(self.light_system.start_sequence)
        pool.apply_async(self.dance_robots.initialize_pose)
```

```
# Dance Music Playing Core Logic
def dance_session(self, music_list):
    conductor = ConductorAI(tempo_detect_mode="real-time")
    for piece in music_list:
        # Dynamically generate band score
        score = MelodyComposer().arrange(
            piece,
            orchestra_size=32,
            complexity=0.8
        )

        # Command robot action generation
        conductor.load_score(score)
        beat_pattern = conductor.analyze_beat()

        # Multithreaded execution
        processes = [
            multiprocessing.Process(target=self.orchestra.play, args=(score,)),
            multiprocessing.Process(target=self.dance_robots.perform,
            args=(beat_pattern, "waltz"))
```